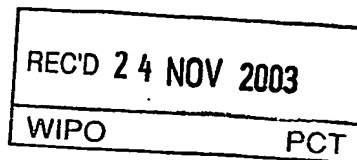PCT/EP03/03844
18.09.03

**Europäisches
Patentamt**

**European
Patent Office**

**Office européen
des brevets**

Bescheinigung   Certificate   Attestation

Die angehefteten Unterla-
gen stimmen mit der
ursprünglich eingereichten
Fassung der auf dem näch-
sten Blatt bezeichneten
europäischen Patentanmel-
dung überein.

The attached documents
are exact copies of the
European patent application
described on the following
page, as originally filed.

Les documents fixés à
cette attestation sont
conformes à la version
initialement déposée de
la demande de brevet
européen spécifiée à la
page suivante.

Patentanmeldung Nr.   Patent application No.   Demande de brevet n°

02022042.2

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Europäisches Patentamt
European Patent Office
Office européen des brevets

Anmeldung Nr:
Application no.:   02022042.2
Demande no:

Anmeldetag:
Date of filing: 01.10.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

SAP Aktiengesellschaft
Neurottstrasse 16
69190 Walldorf
ALLEMAGNE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

XML interfaces in unified rendering

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s)
revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F9/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of
filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

XML interfaces in unified rendering

The invention relates in general to programmable
5 electronic multipurpose computers.

Scripting languages, such as JavaScript and Perl, can
be used to program applications to be run on computer
systems. Correctness of the programmed scripting code
10 is desirable, as with any computer programming in
general. A program can be checked or validated prior to
use to reduce errors occurring during runtime of the
code. It is an object of the invention to reduce the
number of errors in computer code. Therefore, the
15 invention provides for a computer implemented method
for validation of computer code according to claim 1.
By validating both sets of instructions with the script
code used, the number of errors will be reduced.

20 Objects, aspects and advantages of the invention will
be better understood from the following detailed
description of a preferred embodiment of the invention.

A computer program can be defined in design
25 documentation and specifications. From this starting
point an actual implementation can be coded using a
suitable programming language.

According to the invention the coding is done using a
30 two-component interface-classes model, and a script
coded section. Non-limiting examples of such
programming languages are compiler languages and object
oriented programming languages. In compiler languages,
such as for example Modula-2, these components are
35 implemented as definition modules and implementation
modules. In object oriented programming languages, such
as for example Java, C++, C#, and Modula-3, the

- 1 -

components are implemented as interfaces and classes,
wherein interfaces are equivalent to definition modules
and classes are equivalent to implementation modules.
The script-coded section can be programmed using any
5 suitable script language, such as for example
JavaScript and Perl.

Interfaces in object-oriented programming are used for
several purposes. To function for example as a record
10 of promises (with respect to functionality) given by
one ("provider") class. This fact is used to verify
automatically during compilation whether a second
"customer" object is relying on features of a class
that have not been promised. Also the interface is used
15 to verify whether the actual implementation of the
provider class keeps all its promises.

This allows inspection of the code during compilation
and to test or validate whether errors with respect to
20 these fundamentals would occur during runtime. For
example methods that are offered by the object can be
checked, as well as compliance of object classes with
the respective interfaces. If errors are found these
can be reported during compilation, so that the errors
25 in the source code can be corrected. Thus runtime
errors are reduced in the compiled code.

For example, if an interface promises the availability
of a certain method but the method is not present in
30 the implementation, a "customer" of that method would
fail at latest at run time. Interfaces enable modern
compiler languages to verify the availability of the
method during compilation, eliminating this failure
mode with the associated debugging effort.

35

Another example is that can be checked whether a method
of an object is called in the code that is not present

in the interface. This would lead in the compiled code
during runtime to errors. By checking the interface,
this can be caught at the outset, eliminating this
failure mode during runtime.

5

In one embodiment of the invention, a program is
programmed in object oriented style using two separate
tree structures, written in XML, wherein the first tree
structure represents the classes to be implemented and
10 the second tree structure represents the associated
interfaces. Note that the invention is not limited to
the implementation shown in this example, and that any
programming implementation yielding a set of
classes/definitions and interfaces/implementations can
15 be employed. Based on this tree structures in XML
executable programs can be generated, for example using
Extensible Stylesheet Language (XSL) or Apache
Velocity. XSL defines the code using two parts; a
language for transforming XML documents, and an XML
20 vocabulary for specifying formatting semantics. An XSL
style sheet specifies the presentation of a class of
XML documents by describing how an instance of the
class is transformed into an XML document that uses the
formatting vocabulary. On the XML level a syntax check
25 is performed between the interface description and the
implementation class description.

Using this method executable programs can be generated
for different platforms, for example ABAP, .Net and
30 JavaScript. In case the executable code generated does
implement classes and interfaces, such as is the case
with ABAP and .Net, the validity of that code can be
verified with a compiler program that performs the
usage and implementation checks at the semantics level.
35 Then executable program code is generated from the
interface description and from the implementation class

- 3 -

description. The thus generated code can then use or call the script coded section when needed.

5 In this example, the generation of executable programs in script languages, such as for example JavaScript is done by first generating an intermediate code, using a language that supports classes and interfaces. In this example, the intermediate code used can be Java, with implementation for interfaces and classes. The validity

10 of these interfaces and classes is then proved using methods as described above, i.e. using a compiler that performs the usage and implementation checks at the semantics level. This can be done for example by comparing the resulting interfaces with the classes

15 obtained. A successful validation of the Java code means that the original XML code is correct for at least the interface and classes definition. Further the script code section is checked by running it trough an interpreter. This can be for example a JavaScript

20 interpreter if the script code is written in JavaScript. Such interpreters or parsers are known per se in the art. The interpreter yields at least a symbol table including various elements, such as for example variables, used by the script code. The information

25 included in the symbol table is compared to the original XML implementation description. From this comparison can be derived whether the script code is compatible with the implementation description. If the script code is validated accordingly, executable code

30 generated from the sets of definition and implementation instructions, is validated for use with the script code. Thus validated, the number of runtime errors is reduced.

35 In a further embodiment of the invention, two separate tree structures are provided for, together with a script code section. The first tree structure describes

- 4 -

interfaces, while the second tree structure describes content in the form of classes for a HTML document. The script code section provides functionality for the HTML document, such as for example scrolling of text. The

5   tree structures can be programmed for example in XML, and the script code section can be programmed in JavaScript.

The invention further relates to a program storage

10 device readable by a computer system, embodying a program of instructions executable by the computer system to perform any method according to the invention. As this invention may be embodied in several forms without departing from the spirit of essential

15 characteristics thereof, the present embodiment is therefore illustrative and not restrictive, since the scope of the invention is defined by the appended claims rather than by the description preceding them, and all changes that fall within the metes and bounds

20 of the claims, or equivalence of such metes and bounds thereof are therefore intended to be embraced by the claims.

Claims

1. Computer implemented method for validation of computer code, comprising

   providing a computer program by defining at least one set of definition instructions, at least one set of implementation instructions and at least a script code section,

   validating said defined sets using a validation tool, and

   validating said script code section using said set of implementation instructions.

2. Method according to claim 1, wherein the definition instructions are classes and the implementation instructions are interfaces.

3. Method according to claim 1, wherein the definition instructions are converted into classes and the implementation instructions are converted into interfaces.

4. Method according to claim 3, wherein said definition instructions and said implementation instructions are described in XML.

5. Method according to claim 4, wherein said classes and interfaces are defined in Java language.

6. Method according to any of the preceding claims, wherein at least one of said sets of instructions is defined in a tree structure.

7. Method according to any of the preceding claims, wherein said script code section is JavaScript.

- 6 -

8. Method according to any of the preceding claims, wherein validating said script code section comprises generating a symbol table by executing said code section in an interpreter, and comparing said symbol table with the implementation instructions.

9. A program storage device readable by a computer system, embodying a program of instructions executable by the computer system to perform a method according to any of claims 1-8.

2002-223-E

Abstract

Computer implemented method for validation of computer code, comprising

5 providing a computer program by defining a set of definition instructions, a set of implementation instructions and a script code section, validating the defined sets using a validation tool, validating the script code section using the set of implementation

10 instructions, wherein the definition instructions are classes and the implementation instructions are interfaces, wherein the definition instructions are converted into classes and the implementation instructions are converted into interfaces, wherein the

15 definition instructions and said implementation instructions are described in XML.

- 8 -